

What is claimed is:

1 1. A method of ordering program code in a computer memory, the method
2 comprising:
3 selecting an ordering from among a plurality of orderings for a plurality of
4 program code segments using a heuristic algorithm; and
5 ordering the plurality of program code segments in a memory of a
6 computer using the selected ordering.

1 2. The method of claim 1, wherein the heuristic algorithm is configured to
2 minimize cache misses in the computer.

1 3. The method of claim 1, wherein the heuristic algorithm comprises a simulated
2 annealing algorithm.

1 4. The method of claim 3, wherein selecting the ordering using the heuristic
2 algorithm includes testing a subset of the plurality of orderings.

1 5. The method of claim 4, wherein testing the subset of the plurality of orderings
2 includes, for each ordering in the subset, calculating a cost for such ordering based upon
3 cache miss rates for such ordering.

1 6. The method of claim 5, wherein calculating the cost for each ordering
2 comprises calculating a plurality of hits/reference values, misses/address values, and
3 misses/entry values.

1 7. The method of claim 5, wherein testing the subset of orderings includes
2 randomly selecting a different ordering after testing an ordering from the subset of
3 orderings.

1 8. The method of claim 7, wherein randomly selecting the different ordering
2 comprises swapping two program code segments in a previous ordering.

1 9. The method of claim 8, wherein the program code segments each comprise a
2 module, and wherein randomly selecting the different ordering further comprises
3 constraining selection of the two program code segments to modules in the same
4 replaceable unit destination.

1 10. The method of claim 3, wherein selecting an ordering from among the
2 plurality of orderings comprises testing a subset of orderings at each of a plurality of
3 temperature values.

1 11. The method of claim 10, wherein selecting an ordering from among the
2 plurality of orderings further comprises testing a subset of orderings at each temperature
3 value.

1 12. The method of claim 11, wherein selecting an ordering from among the
2 plurality of orderings further comprises accepting a change to an ordering if a calculated
3 cost for such ordering is lower than that of a working ordering.

1 13. The method of claim 11, wherein selecting an ordering from among the
2 plurality of orderings further comprises randomly accepting a change to an ordering even
3 if the calculated cost for such ordering is not lower than that of the working ordering.

1 14. The method of claim 11, wherein selecting an ordering from among the
2 plurality of orderings further comprises prematurely halting the testing of orderings based
3 upon a halt criterion.

1 15. The method of claim 1, wherein the program code segments each comprise a
2 module from an operating system kernel.

1 16. The method of claim 15, wherein each module comprises a high use module,
2 and wherein selecting the ordering from among a plurality of orderings comprises
3 generating a high use module list.

1 17. An apparatus, comprising:

2 a processor; and

3 program code configured to be executed by the processor to optimize
4 execution of program code in a computer of the type including a multi-level
5 memory architecture by using a heuristic algorithm to select an ordering from
6 among a plurality of orderings for a plurality of program code segments in the
7 program code.

1 18. The apparatus of claim 17, wherein the heuristic algorithm is configured to
2 minimize cache misses in the computer.

1 19. The apparatus of claim 17, wherein the heuristic algorithm comprises a
2 simulated annealing algorithm.

1 20. The apparatus of claim 19, wherein the program code is configured to select
2 the ordering using the heuristic algorithm by testing a subset of the plurality of orderings,
3 and wherein the program code is configured to test the subset of the plurality of orderings
4 by, for each ordering in the subset, calculating a cost for such ordering based upon cache
5 miss rates for such ordering.

1 21. The apparatus of claim 20, wherein the program code is configured to test the
2 subset of orderings by randomly selecting a different ordering after testing an ordering
3 from the subset of orderings.

1 22. The apparatus of claim 21, wherein the program code is configured to
2 randomly select the different ordering by swapping two program code segments in a
3 previous ordering.

1 23. The apparatus of claim 22, wherein the program code segments each
2 comprise a module, and wherein the program code is configured to randomly select the
3 different ordering by constraining selection of the two program code segments to modules
4 in the same replaceable unit destination.

1 24. The apparatus of claim 19, wherein the program code is configured to select
2 an ordering from among the plurality of orderings by testing a subset of orderings at each
3 of a plurality of temperature values, and testing a subset of orderings at each temperature
4 value.

1 25. The apparatus of claim 24, wherein the program code is configured to select
2 an ordering from among the plurality of orderings by accepting a change to an ordering if
3 a calculated cost for such ordering is lower than that of a working ordering.

1 26. The apparatus of claim 25, wherein the program code is configured to select
2 an ordering from among the plurality of orderings by randomly accepting a change to an
3 ordering even if the calculated cost for such ordering is not lower than that of the working
4 ordering.

1 27. The apparatus of claim 17, wherein the program code is configured to select
2 an ordering from among the plurality of orderings by prematurely halting the testing of
3 orderings based upon a halt criterion.

1 28. The apparatus of claim 17, wherein the program code segments each
2 comprise a module from an operating system kernel.

1 29. The apparatus of claim 28, wherein each module comprises a high use
2 module, and wherein the program code is configured to select the ordering from among a
3 plurality of orderings by generating a high use module list.

4 30. A program product, comprising:

5 first program code configured to optimize execution of second program
6 code in a computer of the type including a multi-level memory architecture by
7 using a heuristic algorithm to select an ordering from among a plurality of
8 orderings for a plurality of program code segments in the second program code;
9 and

10 a signal bearing medium bearing the first program code.

1 31. The program product of claim 30, wherein the signal bearing medium

2 includes at least one of a recordable medium and a transmission medium.